HPCDATAMGM-2044 – Registration w/ link to existing data in S3 Archive

Design Document

Version 1.0

Date

# Version History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version Number | Implemented By | Revision Date | Approved By | Approval Date | Description of Change |
| 1.0 | Eran Rosenberg | 2/19/2025 |  |  | Initial Document |

Table of Contents

[Version History 2](#_Toc179816669)

[1 Introduction 4](#_Toc179816670)

[1.1 Purpose of this document 4](#_Toc179816671)

[1.2 Background 4](#_Toc179816672)

[1.3 Scope 4](#_Toc179816673)

[1.4 Assumptions 4](#_Toc179816674)

[1.5 Constraints 4](#_Toc179816675)

[2 Architecture Design 4](#_Toc179816676)

[2.1 Hardware Architecture 4](#_Toc179816677)

[2.2 Software Architecture 4](#_Toc179816678)

[3 Software Design 4](#_Toc179816679)

[3.1 Introduction 4](#_Toc179816680)

[3.2 Detailed Design 4](#_Toc179816681)

[3.3 User Interface Design 4](#_Toc179816682)

[3.4 System Integration 4](#_Toc179816683)

[4 Database Design 4](#_Toc179816684)

[4.1 Table Design 4](#_Toc179816685)

# Introduction

## Purpose of this document

This document provides a design for HPCDATAMGM-2044. DME users indicated a need to register files that already exist in S3 archive. Currently, all file registrations include a data transfer upload of user’s source (Globus, S3, Google Drive, etc). With this new capability, the files will be registered in iRODs, but the data-transfer action will be skipped, and instead the registered file will be ‘linked’ (via system metadata) to the location of the files in the target S3 archive

## Background

## Scope

### DME Posix archive is out of scope. Only S3 archive is in scope

## Assumptions

## Constraints

# Architecture Design

## Hardware Architecture

## Software Architecture

# Software Design

## Introduction

## Detailed Design

### API

A new API to scan an archive will be created.

Endpoint: /linkArchive

#### Sample JSON Request

{

"dryRun": true,

"directoryScanRegistrationItems": [

{

"basePath": "/TEST\_Archive/PI\_Collection",

"archiveScanDirectory": {

"directoryLocation": {

"fileContainerId": "vast-bucket",

"fileId": "/existing-path-to-scan"

}

},

"s3ArchiveConfigurationId" : "vast-s3-archive-config-id",

"includePatterns": [],

"excludePatterns": []

}

]

}

#### Sample JSON Response

{

"dataObjectRegistrationItems": [

{

"path": "/TEST\_Archive/PI\_Collection/some-file",

"archiveLinkSource": {

"sourceLocation": {

"fileContainerId": " vast-bucket",

"fileId": existing-path-to-scan/some-file"

}

},

"s3ArchiveConfigurationId": " vast-s3-archive-config-id "

}

]

}

### API Implementation

#### Request Validation –HpcDataManagementServiceImpl::validateDataObjectRegistrationRequest()

#### This method is responsible to validate a data object registration request. It will need to be extended to validate a registration w/ archive linking. One of the validations it will need to perform is that the bucket provided by the user in the request matches the bucket configured with the s3-archive-configuration-id also provided by the user.

/\*\*

\* Validate a data object registration request (that is part of bulk

\* registration)

\*

\* **@param** registrationRequest The registration request.

\* **@param** path The registration path.

\* **@throws** HpcException If the request is invalid.

\*/

**private** **void** validateDataObjectRegistrationRequest(HpcDataObjectRegistrationRequest registrationRequest,

String path) **throws** HpcException {

#### Archive Scanning – HpcDataTransferService::scanDirectory() The implementation will need to scan the provided archive bucket and folder to identify files to register and link to. We already have an application service that supports scanning an S3 bucket. It already supports scanning a S3 bucket owned by the user OR an S3 archive (configured via S3-archive-configure-id). We will use this app service as is to scan the archive folder and generate the list of files found and listed in the response to the API. Here is the documentation of the service for reference /\*\*

\* Scan a directory (recursively) and return a list of all its files.

\*

\* **@param** dataTransferType (Optional) The data transfer type. If null, a

\* directory on the local DME server NAS is

\* scanned.

\* **@param** s3Account (Optional) S3 account to use.

\* **@param** googleAccessToken (Optional) Google Drive/Storage access-token

\* to use.

\* **@param** directoryLocation The endpoint/directory to scan and get a list

\* of files for.

\* **@param** configurationId The configuration ID (needed to determine the

\* archive connection config).

\* **@param** s3ArchiveConfigurationId (Optional) The S3 Archive configuration ID.

\* Used to identify the S3 archive the

\* data-object is stored in. This is only

\* applicable for S3 archives, not POSIX.

\* **@param** includePatterns The patterns to use to include files in the

\* scan results.

\* **@param** excludePatterns The patterns to use to exclude files from the

\* scan results.

\* **@param** patternType The type of the patterns provided.

\* **@return** A list of files found.

\* **@throws** HpcException on service failure.

\*/

**public** List<HpcDirectoryScanItem> scanDirectory(HpcDataTransferType dataTransferType, HpcS3Account s3Account, String googleAccessToken, HpcFileLocation directoryLocation, String configurationId,String s3ArchiveConfigurationId, List<String> includePatterns, List<String> excludePatterns, HpcPatternType patternType) **throws** HpcException;

#### Creating a bulk registration task If the ‘dryRun’ attribute is set to false, the API implementation will create a bulk data object registration task that includes all the items identified in the scan. The implementation will reuse the following DAO method as is /\*\*

\* Store a new bulk data object registration task (if

\* dataObjectListRegistrationTask.getId() is provided NULL), or update an

\* existing task. Note: If a new task is inserted,

\* dataObjectDownloadTask.getId() will be updated with the generated ID.

\*

\* **@param** dataObjectListRegistrationTask The data object registration task to

\* persist.

\* **@throws** HpcException on database error.

\*/

**public** **void** upsertBulkDataObjectRegistrationTask(HpcBulkDataObjectRegistrationTask dataObjectListRegistrationTask) **throws** HpcException;

### Scheduled Task Implementation

#### The scheduled task processBulkDataObjectRegistrationTasks() handling the bulk registration requests. It is calling DataMagaementBusService::registerDataObject() to perform the registration.

#### HpcDataManagementBusService::registerDataObject() implementation will need to be modified the following – right after completing the iRODS registration and user metadata are written to iRODS, the service will need to either continue the existing logic and issue a data upload request, or in the case of archive linking – call a new method HpcDataManagementBusService::linkArchiveSource() to link the data object in iRODs with the existing file in that S3 archive

#### HpcDataManagementBusService::linkArchiveSource() The implementation is very close to existing service to complete singlepart upload - /\*\*

\* Complete an upload from S3 (either via URL upload or streaming). Checks if

\* the data object is present in the S3 archive and update the data management

\* system accordingly.

\*

\* **@param** path The path of the data object to check if an

\* upload from S3 completed.

\* **@param** systemGeneratedMetadata The system generated metadata for the data

\* object.

\* **@return** true if the uploaded completed, or false otherwise (i.e. the data

\* object upload is still in progress).

\* **@throws** HpcException If failed to check/update upload status.

\*/

**public** **boolean** completeS3Upload(String path,

HpcSystemGeneratedMetadata systemGeneratedMetadata)

**throws** HpcException;

##### The implementation will check that the file exists in the archive. However, if the file in the archive is already linked to another file in iRODS, it will error out

## User Interface Design

## System Integration

# Database Design

## Table Design

No DB changes